

How to use Portal Session Notification with your own custom applications

Applicable Releases:

SAP Netweaver 7.0

SAP Netweaver Composition Environment 7.1 and 7.2

IT Practice:

User Productivity Enablement

IT Scenario:

Running an Enterprise Portal

Version 1.0

September 2009



© Copyright 2009 SAP AG. All rights reserved.

No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

Some software products marketed by SAP AG and its distributors contain proprietary software components of other software vendors.

Microsoft, Windows, Outlook, and PowerPoint are registered trademarks of Microsoft Corporation.

IBM, DB2, DB2 Universal Database, OS/2, Parallel Sysplex, MVS/ESA, AIX, S/390, AS/400, OS/390, OS/400, iSeries, pSeries, xSeries, zSeries, z/OS, AFP, Intelligent Miner, WebSphere, Netfinity, Tivoli, Informix, i5/OS, POWER, POWER5, OpenPower and PowerPC are trademarks or registered trademarks of IBM Corporation.

Adobe, the Adobe logo, Acrobat, PostScript, and Reader are either trademarks or registered trademarks of Adobe Systems Incorporated in the United States and/or other countries.

Oracle is a registered trademark of Oracle Corporation.

UNIX, X/Open, OSF/1, and Motif are registered trademarks of the Open Group.

Citrix, ICA, Program Neighborhood, MetaFrame, WinFrame, VideoFrame, and MultiWin are trademarks or registered trademarks of Citrix Systems, Inc.

HTML, XML, XHTML and W3C are trademarks or registered trademarks of W3C®, World Wide Web Consortium, Massachusetts Institute of Technology.

Java is a registered trademark of Sun Microsystems, Inc.

JavaScript is a registered trademark of Sun Microsystems, Inc., used under license for technology invented and implemented by Netscape.

MaxDB is a trademark of MySQL AB, Sweden.

SAP, R/3, mySAP, mySAP.com, xApps, xApp, SAP NetWeaver, and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP AG in Germany and in several other countries all over the world. All other product and service names mentioned are the trademarks of their respective companies. Data contained in this document serves informational purposes only. National product specifications may vary.

These materials are subject to change without notice.

These materials are provided by SAP AG and its affiliated companies ("SAP Group") for informational purposes only, without representation or warranty of any kind, and SAP Group shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP Group products and services are those that are set forth in the express warranty statements accompanying such products and services, if any. Nothing herein should be construed as constituting an additional warranty.

These materials are provided "as is" without a warranty of any kind, either express or implied, including but not limited to, the implied warranties of merchantability, fitness for a particular purpose, or non-infringement.

SAP shall not be liable for damages of any kind including without limitation direct, special, indirect, or consequential damages that may result from the use of these materials.

SAP does not warrant the accuracy or completeness of the information, text, graphics, links or other items contained within these materials. SAP has no control over the information that you may access through the use of hot links contained in these materials and does not endorse your use of third party web pages nor provide any warranty whatsoever relating to third party web pages.

SAP NetWeaver "How-to" Guides are intended to simplify the product implementation. While specific product features and procedures typically are explained in a practical business context, it is not implied that those features and procedures are the only approach in solving a specific business problem using SAP NetWeaver. Should you wish to receive additional information, clarification or support, please refer to SAP Consulting.

Any software coding and/or code lines / strings ("Code") included in this documentation are only examples and are not intended to be used in a productive system environment. The Code is only intended better explain and visualize the syntax and phrasing rules of certain coding. SAP does not warrant the correctness and completeness of the Code given herein, and SAP shall not be liable for errors or damages caused by the usage of the Code, except if such damages were caused by SAP intentionally or grossly negligent.

Disclaimer

Some components of this product are based on Java™. Any code change in these components may cause unpredictable and severe malfunctions and is therefore expressly prohibited, as is any decompilation of these components.

Any Java™ Source Code delivered with this product is only to be used by SAP's Support Services and may not be modified or altered in any way.

Document History

Document Version	Description
1.00	First official release of this guide

Typographic Conventions

Type Style	Description
<i>Example Text</i>	Words or characters quoted from the screen. These include field names, screen titles, pushbuttons labels, menu names, menu paths, and menu options. Cross-references to other documentation
Example text	Emphasized words or phrases in body text, graphic titles, and table titles
Example text	File and directory names and their paths, messages, names of variables and parameters, source text, and names of installation, upgrade and database tools.
Example text	User entry texts. These are words or characters that you enter in the system exactly as they appear in the documentation.
<Example text>	Variable user entry. Angle brackets indicate that you replace these words and characters with appropriate entries to make entries in the system.
EXAMPLE TEXT	Keys on the keyboard, for example, F2 or ENTER.

Icons





Icon	Description
	Caution
	Note or Important
	Example
	Recommendation or Tip

Table of Contents

1.	Business Scenario	1
2.	Background Information	2
2.1	Request flow of the Session Release Notification.....	2
2.2	Dealing with Popup Blockers in the Browser.....	3
2.3	Request Distribution via Server side connections.....	4
2.4	SAP Internet Session Protocol.....	4
2.5	Parameters of the termination URL.....	5
2.5.1	Explanation of the parameters:.....	5
2.6	Logging and Monitoring.....	6
2.6.1	Logging.....	6
2.6.2	Monitoring.....	7
3.	Prerequisites	8
4.	Step-by-Step Procedure	9
4.1	Create a Portal Application and a Portal Component.....	9
4.2	Create a UUID helper class.....	10
4.3	Create the Custom Portal Component.....	10
4.4	Deploy and test your portal component.....	12
4.5	Create an external test application.....	12

1. Business Scenario

One reason to use the SAP Enterprise Portal is the easy integration of state full (existing) Web Applications into the Portal.

When integrating state-full applications you always face the problem, that the application is not notified if the user navigates away or even closes the browser window. All Web Applications face this problem, since they are based on the stateless HTTP protocol. Nearly all state full applications keep session data on the server side and release those resources after a predefined timeout. The servlet container for example releases session data by default after 30 minutes inactivity. As a consequence there is a delay between the user action (e.g. the user has closed the browser) and the release of the corresponding session data on the server side. This delay can cause the following problems:

- “Dead State” remains on the server and blocks resources (memory) until timeout.
- Server scalability and performance get worse due to the load which is caused by this “Death State”
- Application locks may lead to deadlocks, when the same or different users accessing the same shared resources.

If you use SAP technology to build your Web Applications and run them in the portal, they automatically include a feature called **Session Release Notification**, which informs the external content server if the user

- navigates away
- closes the browser window
- logs-off the portal

The following SAP technologies use this Session Release Notification out of the box without the need to do/implement anything.

- ITS based applications
- BSP (Business Server Pages)
- Web Dynpro (ABAP and Java)

However if you have build your applications using non SAP technologies, your content server will not get notified automatically.

This Ho-to guide shows how you integrate external applications into the portal which will be notified when the user navigates away closes the browser or logs-off the portal.

2. Background Information

2.1 Request flow of the Session Release Notification

The scenario described here is based on the assumption, that the external application is not running inside the portal runtime (is not a Portal Component).

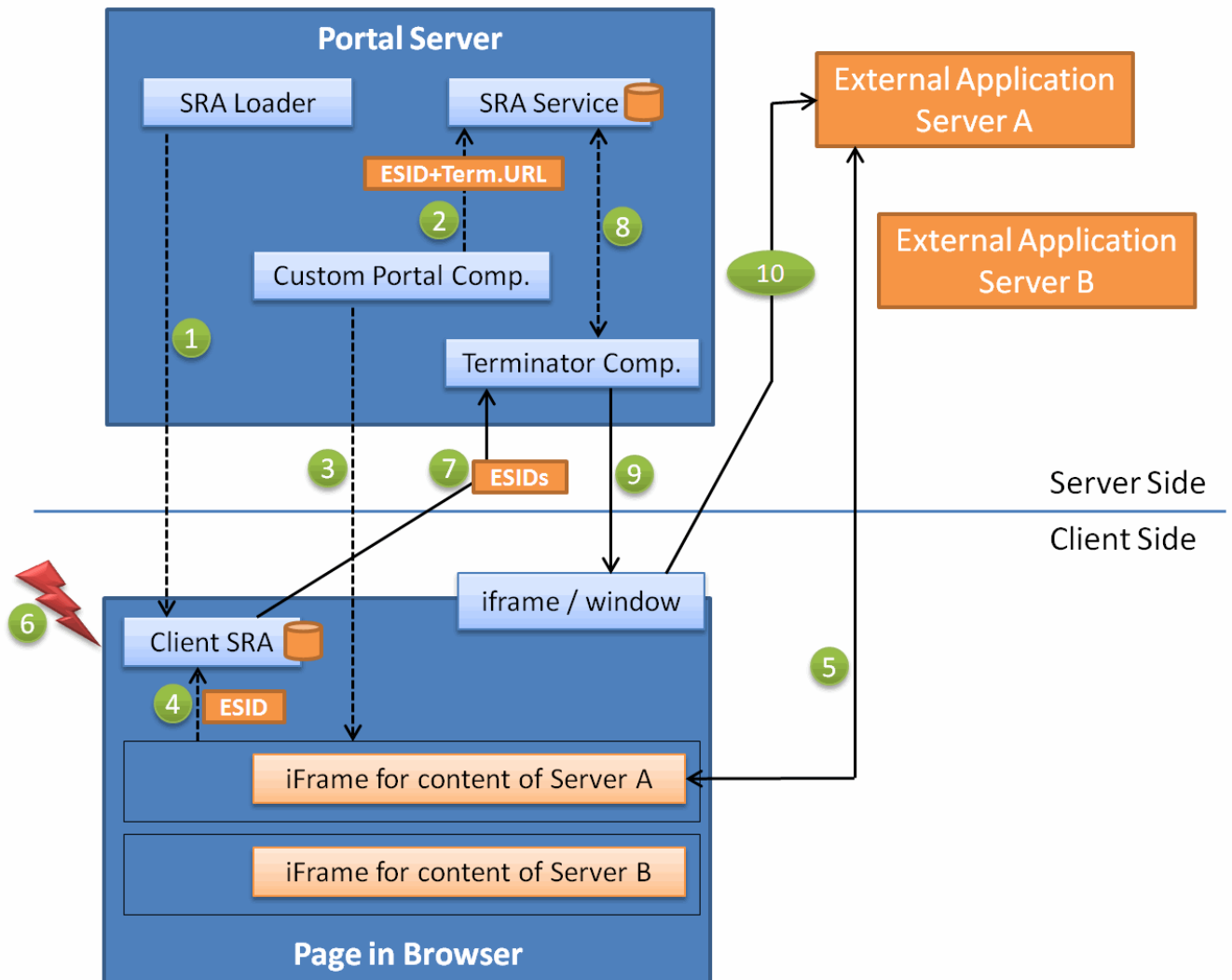


Figure 1: Request Flow

- 1) The SRA Loader (Session Release Agent) is a portal component, which takes care of including the necessary Java Script code of the Client SRA into the HTTP response of portal pages or iViews. The Client SRA is a Java Script object which is responsible for collecting the session relevant data from the applications into his own storage. The lifetime of the Client SRA object is related to the lifetime of the portal page or browser window.
- 2) For the Session Release Notification, you need to develop your own custom portal component. The applications you want to integrate into your portal must be based on this portal component (create new iViews from this portal component).

The custom portal component creates a unique key, which is used as an “External Session ID” (ESID). The ESID and additional session information like the termination URL is then stored into the SRA Service. The termination URL is the URL which will later be called to notify the external application that the user for example closed the browser window or navigated away. The termination URL also contains the ESID as a parameter. An example of the termination URL could be:

`http://server:port/ApplicationA/termination/?sap-esid=3fa953c643`

- 3) The custom portal component now builds the HTTP response. It creates an iframe tag with the URL to the external application and adds the ESID as a parameter to this iframe URL.
Example of the iframe:
`<iframe src="http://server:port/ApplicationA/?sap-esid=3fa953c643" />`
The custom portal component also adds some Java Script code in the response, which registers/stores the ESID into the Client SRA object
- 4) The Java Script code is executed at the client. It stores the ESID into the Client SRA object.
- 5) The iframe URL with the ESID as parameter is now called from the browser. The external application must then map this ESID to its own session ID. Later if the user for example closes the browser window, the termination URL with the ESID is send to the external application. The external application must then read the mapping, get its own session ID and closes the session.
- 6) If the portal page is now destroyed (user navigates away, logs-off or closes the browser) the Client SRA wakes up like a WatchDog (triggered by the onunload browser event).
- 7) The Client SRA sends the collected ESIDs (in case of multiple iframes on one page) to a portal component called Terminator.
- 8) The Terminator Component reads the session info (especially the termination URL) from the SRA Service.
- 9) The HTTP response of the Terminator Component goes back to the browser. The response includes the termination URL with the ESID parameter. In order to fire this termination URL to the external application, we need a stable window to send the termination URL. This is achieved by using a hidden window or a hidden iframe.
- 10) The hidden window or iframe makes a new request to the external application including the ESID as parameter. The external application now has to look up her own session ID with the ESID and can then close and release her own sessions/resources.

2.2 Dealing with Popup Blockers in the Browser

Newer version of browsers (Mozilla Firefox, MSIE ...) or even the browser plug-ins or toolbars (Google Toolbar, Yahoo Toolbar) provides the Browser popup blockers, which may cause trouble in relation to SRA feature.

In order to handle termination notification when the current window is being destroyed (typically logoff or closing this window by user), we need a stable window to accomplish the network communication. If the popup blocker avoids opening such windows, then this may lead to malfunctioning of SRA.

Therefore, whenever the portal runs in the browser with installed popup blocker, you have to make sure that this Popup blocker is deactivated for those URLs that point to the portal server or content server used in the scenario. Generally you will have to put the hostnames or URLs to the “white list”, just to allow the opening of new windows from such pages. Check the documentation for your Popup blocker how this can be achieved.

2.3 Request Distribution via Server side connections

In the previously described scenario the termination URL is send from the client, either from a hidden frame or window (see step 10 in figure 1). In order to minimize the network traffic between browser and backend, one can consider the possibility to send the termination requests from the Terminator component directly via server side URL connection to the external application (see the following figure).

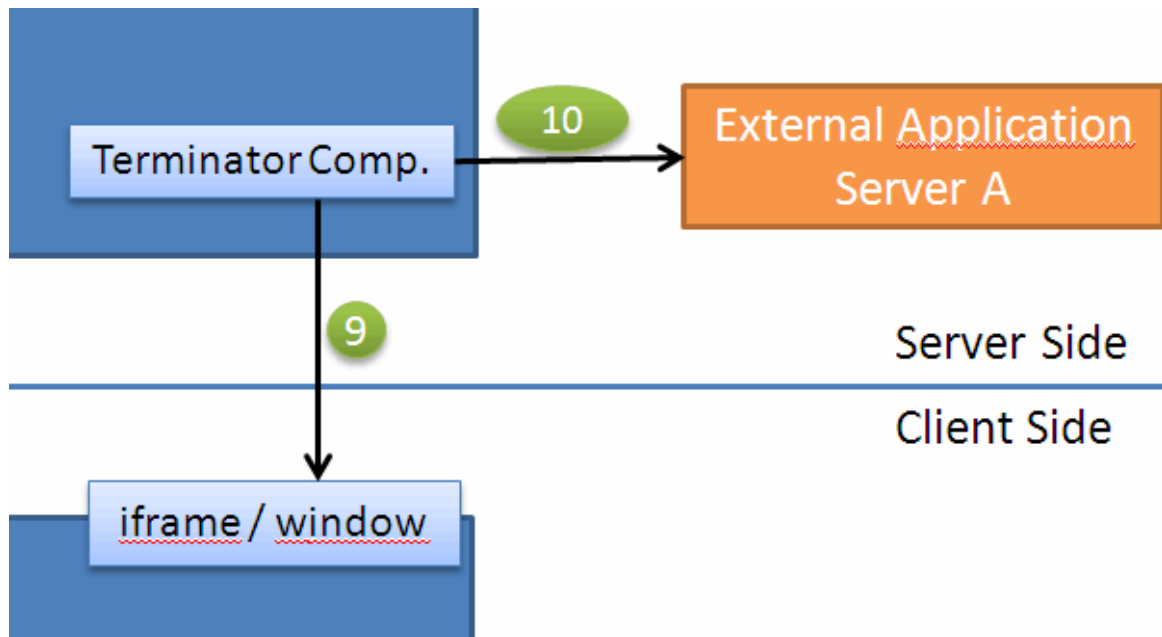


Figure 2: Server side connections

Even though the request distribution via server connection reduces the count of requests, this can't be used as the default solution. It may cause problems when the content server runs HTTPS, uses client side certificates (content server avoids in such case the connection from the server side) or for some cases with load-balancing. Using server to server communication could also result in configuration tasks for firewalls/proxies.

The Terminator component in the portal supports and combines both these methods. By default the request distribution via client is used, but can be switch to distribution via server connections by a corresponding setting on the portal service (see *Terminator.sendMode*). SAP note #765914 describes exactly how to configure the send mode.

2.4 SAP Internet Session Protocol

Starting with SAP Workplace 2.x, all content servers, that want to participate on the Session Release Notification, must understand "SAP Internet Session Protocol" (SISP).

There are currently 3 versions of the protocol, 1.0, 1.1 and 2.0. Version 2.0 was introduced with NW Portal 7.0 SPS 5. The explanation of the protocol is out of scope of this document. The most important thing is, that the different protocol versions support different session commands send to the external applications (explanations of session commands follow).

Version 1.0 of SISP is quite old, has a lot of technical problems and should not be used anymore!

2.5 Parameters of the termination URL

The termination URL which is send from the browser (or from the terminator component) to the external application contains the following parameters:

Request Parameter Name	Possible Values
sap-esid	ESID = A unique identifier
sap-sessioncmd	USR_ABORT, USR_LOGOFF, USR_SUSPEND
~SAPSessionCmd	USR_ABORT, USR_LOGOFF, USR_SUSPEND
SAPWP_ACTIVE	1
dsmguid	Numerical timestamp of the action

2.5.1 Explanation of the parameters:

sap-esid: This parameter is added by your own custom portal component. All other following parameters are added by the SRA framework automatically. The external application uses this ESID value to get its mapped own session ID.

sap-sessioncmd and ~SAPSessionCmd: The session command has to be passed twice because of the backwards compatibility. The parameter values indicate what the user has done. Depending on the user action, the external application can behave differently.

The following table shows the mapping of the user action to the session command depending on the SISP protocol version.

User action / Protocol Version	SISP 1.1	SISP 2.0
User is logging out from the portal	USR_LOGOFF	USR_LOGOFF
Closing an Browser Window or any action which displaces the application (but not the portal navigation !)	USR_ABORT	USR_ABORT
Portal Navigation, where application in the Workarea-frame is replaced with different one.	USR_ABORT	USR_SUSPEND
Portal Navigation , where application Workarea-frame is	USR_ABORT	no notification !

replaced with the same one – Means we are re-launching the existing iVlew/application instance		
---	--	--

SISP 2.0 has a new session command (USR_SUSPEND), which is send to the external application if the user navigates away. In SISP 1.1 you cannot differentiate between a user closing the browser window or navigating away. In both cases USR_ABORT is send.

Another difference between 1.1 and 2.0: When re-launching the same application (detected by the Portal Navigation), the portal SRA will recognize such case for SISP 2.0 and omit the USR_SUSPEND notification for performance reasons (otherwise the content server/external server would have to terminate the existing session and create the new one few milliseconds later).

SAPWP_ACTIVE: the flag signaling the application is running in the portal. You can use this, if the application needs to render different code running within the portal (SAPWP_active=1) and without the portal (undefined or SAPWP_active=0).

dsmguid: The numeric timestamp (current milliseconds since 1.1.1970) is needed because otherwise the browser could resolve the termination URL from the browser cache instead of sending a up to date HTTP request.

2.6 Logging and Monitoring

2.6.1 Logging

Before you can start to look at the log files, you must configure the log controller and increase the log level for the DSM Controller. Perform the following steps to increase the log level:

1. Start the Visual Administrator for the AS Java
2. Open the Log Configurator for the given Server node (*Server xxx -> Services -> Log Configurator*)
3. Search for the DSM Controller using *Runtime/Locations* tab and open the path *com -> sap -> portal -> dsm*
4. Change the severity from **Error** to **All**

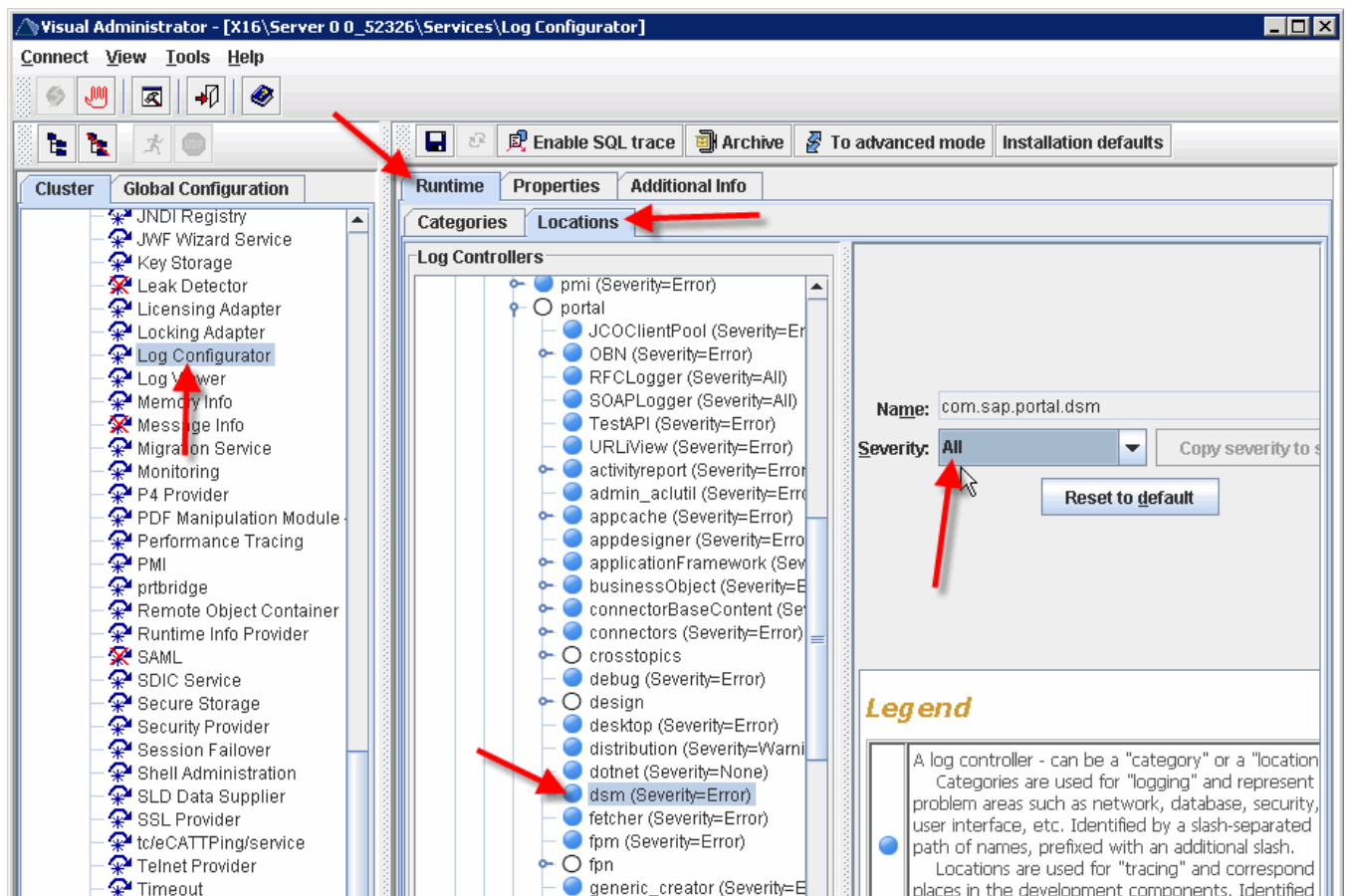


Figure 4: Configuration of Logging

The collected server logging outputs can then be extract from the defaultTrace.trc as follows:

1. Start the standalone log viewer (in Windows based installation, the location is typically under ... \j2ee\admin\logviewer-standalone)
2. Open the *defaultTrace.trc*, (typically under ...->j2ee->cluster->server X->log)
3. When displayed, use the Filter to display on the relevant values

Text: **com.sap.portal.dsm**

In columns: **Location**

and then use button “[] include in Current Log”

2.6.2 Monitoring

The portal offers monitoring and tracing support for the Session Release Notification. You can call the monitoring component to test the elementary API functionality of the Session Release Agent, even not having external content systems connected. The start URL is the following:

<PortalServer>/irj/servlet/prt/portal/prtroot/com.sap.portal.dsm.Monitor

At the top, the user can enter any URL which will be loaded after pressing the “Go” button to the bottom Sandbox frame of this application. Typically you will put there the URL to start i.e. an external application. Whenever the SessionInfoObject or SessionInfoKey is passed, you get these properties listed in the middle part of the application. When pressing the “Terminate all” button, the termination notification is explicitly started in the same way as when the standard onunload action is performed.

On the right side you can navigate to the SessionInfo keys (ESIDs) which are stored on the SRA Service (button "Lookup on Server"). It is also possible manually register an ESID into the SRA Service (button Register Key).

Figure 5: Monitoring Component of SRA

3. Prerequisites

- You have installed a SAP Netweaver Portal Version 7.0
- You have a corresponding SAP Netweaver Developer Studio installed locally
- You are familiar with Java and Portal Development (how to deploy portal components, how to create iViews based on portal components etc.)

4. Step-by-Step Procedure

4.1 Create a Portal Application and a Portal Component

- 1) Create a Portal Application called *SRAWrapper*
- 2) Create a new *Portal Component* of type *AbstractPortalComponent* called *iView*

To be able to use the Session Release Notification you must set a *SharingReference* in your *portalapp.xml* deployment descriptor to the “epcf loader service” and the “dsm service”.

```
<application-config>
    <property name="SharingReference"
        value="com.sap.portal.epcf.loader, com.sap.portal.dsm"/>
</application-config>
```

- 3) To be able to configure the URL to the external application and the termination URL, we add two component properties called “Application_URL” and “Termination_URL”.

```
<component-profile>
    <property name="Application_URL">
        <property name="plainDescription" value="URL to start the
            external Application"/>
        <property name="longDescription" value="URL to start the
            external Application"/>
        <property name="personalization" value="no-dialog"/>
        <property name="category" value="External Application Data"/>
    </property>
    <property name="Termination_URL">
        <property name="plainDescription" value="URL which should be
            called by the SRA service"/>
        <property name="longDescription" value="URL which should be
            called by the SRA service"/>
        <property name="personalization" value="no-dialog"/>
        <property name="category" value="External Application Data"/>
    </property>
</component-profile>
```

- 4) To be able to compile your sourcecode, you must add the following jar files to your classpath:

```
\usr\sap\<SID>\<instance>\j2ee\cluster\server0\apps\sap.com\irj\
servlet_jsp\irj\root\WEB-INF\portal\portalapps\com.sap.portal.dsm\
lib\ com.sap.portal.dsm_api.jar
```

```
\usr\sap\<SID>\<instance>\j2ee\cluster\server0\apps\sap.com\irj\
servlet_jsp\irj\root\WEB-INF\portal\portalapps\
com.sap.portal.epcf.loader\lib\ com.sap.portal.epcf.loader_api.jar
```

```
\usr\sap\<SID>\<instance>\j2ee\cluster\server0\bin\system\ logging.jar
```

4.2 Create a UUID helper class

Since your own portal component must create unique external session IDs (ESIDs), we need a helper class to generate such keys. If you are using JDK 1.5 or above, you can use the class `java.util.UUID` in the standard Java library.

If you are on NW 7.0, you can only use JDK 1.4, and therefore must create your own UUID class. Using this [link](http://www.sdn.sap.com/irj/scn/index?rid=/library/uuid/709c6123-1884-2c10-4eb8-cfee09bbb94) (<http://www.sdn.sap.com/irj/scn/index?rid=/library/uuid/709c6123-1884-2c10-4eb8-cfee09bbb94>), you can see an example UUID class which will be used by the custom portal component (click on the button “View this Tool” to download the class file).

4.3 Create the Custom Portal Component

The following class can also be downloaded via this [link](http://www.sdn.sap.com/irj/scn/index?rid=/library/uuid/20b81220-df84-2c10-d2aa-f01f9aa09d0c) (<http://www.sdn.sap.com/irj/scn/index?rid=/library/uuid/20b81220-df84-2c10-d2aa-f01f9aa09d0c>):

```
public class iView extends AbstractPortalComponent
{
    private static final String PROPERTY_APPLICATION_URL = "Application_URL";
    private static final String PROPERTY_TERMINATION_URL = "Termination_URL";

    public void doContent(IPortalComponentRequest request,
        IPortalComponentResponse response)
    {
        // 1) read the profile
        IPortalComponentProfile profile = request.getComponentContext().getProfile();
        String applicationURL = profile.getProperty(PROPERTY_APPLICATION_URL);
        String terminationURL = profile.getProperty(PROPERTY_TERMINATION_URL);

        // 2) create ESID
        String sap_esid = new UUID().toString();

        // 3) append sap_esid to applicationURL and terminationURL
        applicationURL = appendSAP_ESID(applicationURL, sap_esid);
        terminationURL = appendSAP_ESID(terminationURL, sap_esid);

        // 4) create a session info object and store it in SRA Service
        IDsmService dsmService = (IDsmService) PortalRuntime.getRuntimeResources().
            getService(IDsmService.KEY);
        ISessionInfoFactory sessInfoFactory = dsmService.getSessionInfoFactory();
        ISessionInfo sessInfo = sessInfoFactory.createSessionInfoObject(
            sap_esid, "1.1", "", terminationURL, terminationURL, terminationURL);
        ISessionInfoStore sessInfoStore = dsmService.getSessionInfoStore();
        sessInfoStore.put(request, sap_esid, sessInfo);

        // 5) write the iframe with the application URL
        response.write("\n<iframe src=\"" + applicationURL +
            "\" frameborder=\"0\" name=\"isolatedWorkArea\""+
            " id=\"isolatedWorkArea\" width=\"100%\" "+
            " marginheight=\"0\" marginwidth=\"0\"></iframe>\n");

        // 6) register the ESID into the Client SRA
        response.write("<SCRIPT>EPCM.DSM.registerFullKey("+
            "\"" + sap_esid + "\"");</SCRIPT>\n");

        // 7) make the iframe full height of the content area
        response.write("\n<script>workArea.resizeIframe();\n");
        response.write("EPCM.subscribeEvent("+
            " \"/urn:com.sapportals.portal:browser\", \"resize\", "+
            " workArea.resizeIframe);\n");
        response.write("EPCM.subscribeEvent("+
```

```
        " \\"urn:com.sapportals.portal:browser\"", \\"load\"", "+  
        " workArea.resizeIframe);</script>\n");  
    }  
  
    private String appendSAP_ESID(String url, String sap_esid){  
        if(url.indexOf('?') == -1)  
            url += "?sap-esid="+sap_esid;  
        else  
            url += "&sap-esid="+sap_esid;  
        return url;  
    }  
}
```

Explanation of the source code:

Step 1) read the profile

When the content administrator creates an iView based on this portal component, he can configure the two properties `Application_URL` and `Termination_URL`. The application URL will be used to call the external application inside an iframe. The termination URL will be called by SRA to inform the external application in case the user does a log off, navigates away or closes the browser window.

Step 2) create the ESID

This line creates the external session ID with the help of the UUID class mentioned above

Step 3) append the ESID to the application URL and the termination URL

We now append the generated ESID to the application URL and the termination URL as a request parameter (using '&' or '?' depending on already existing parameters)

Step 4) create a session info object and store it in SRA service

We first get a reference to the DSM service to get a session info factory object. On this object, we create a session info object. The method `createSessionInfoObject` method is overloaded and has several signatures. The signature we are using here defines the following parameters:

1. **Parameter:** The generated ESID
2. **Parameter:** The version of the SISP protocol
3. **Parameter:** A description of the session info (we could use the name of the external application)
4. – 6. **Parameter:** The abort URL, the logoff URL and the suspend URL

As you see, it is also possible to fire different URLs depending on the user action (user closes the browser window, user logs off or user navigates away). In our case the external application should not differentiate between these cases. The external application will always destroy the session.

Step 5) write the iframe using the application URL

In this step the iframe with the corresponding application URL is written to the client. The id and the name of the iframe should be "isolatedWorkArea", since we use standard portal libraries, which allows us to resize the height of the iframe.

Step 6) register the ESID into the Client SRA

This JavaScript will register and store the ESID into the Client SRA object (see step 4 in figure 1)

Step 7) make the iframe full height of the content area

With this JavaScript, the iframe height will be resized dynamically depending on the size of the content area. This prevents for example having two scrollbars (one from the iframe and one from the browser window)

4.4 Deploy and test your portal component

After you have deployed your portal component, you can create an iView out of the portal component (PAR file). All your external applications you want to integrate and which want to use SRA must be based on this portal component.

Using the properties Application_URL and Termination_URL, you can configure your iView to call the external application.

4.5 Create an external test application

As said before, the task of the external application is to map the retrieved ESID to its own session ID, when the application is called the first time. If the termination URL with the ESID and the session command (sap-sessioncmd) is sent, the external application should destroy its session.

- 1) Create an Enterprise Application Project and a Web Module Project
- 2) Create a servlet with the following sourcecode.

```
public class ExtApp extends HttpServlet {

    // Map to store the esid /
    private Map sessMap = Collections.synchronizedMap(new HashMap());

    protected void doGet(HttpServletRequest request,
        HttpServletResponse response) throws ServletException, IOException {

        // 1) create a statefull application => so we create a session
        HttpSession session = request.getSession();

        // 2) read the parameters
        String esid = request.getParameter("sap-esid");
        String sessionCmd = request.getParameter("sap-sessioncmd");
        if(esid != null && sessionCmd == null){
            // 3) the application is called from within the portal
            // => store esid with the session ID
            sessMap.put(esid,session.getId());
        }
        else if(esid != null && sessionCmd != null){
            // 4) the termination URL is send
            sessMap.remove(esid);
            session.invalidate();
            return;
        }

        // 5) create the response, and show the entries in the mapping table
```

```
PrintWriter out = response.getWriter();
out.write("<html><body><h3>Entries in the mapping table</h3>"+
"<table border=\"1\">");
Iterator iter = sessMap.entrySet().iterator();
while (iter.hasNext()) {
    out.write("<tr>");
    Map.Entry entry = (Map.Entry)iter.next();
    out.write("<td>"+entry.getKey()+"</td>");
    out.write("<td>"+entry.getValue()+"</td>");
    out.write("</tr>");
}
out.write("</table></body></html>");
}
}
```

Explanation of the source code

The HashMap is an instance variable of the servlet. Therefore it is important to synchronize it, because several threads will access and modify the map concurrently.

Step 1: create a statefull application

With this line we create a HTTP session.

Step 2: read the parameters

Read the request parameters.

Step 3: the application is called from within the portal

If the application is called from the portal iframe (content request), there is only one SRA parameter available (`sap-esid`). In this case the ESID and the internal session id are stored into the HashMap as key/value pair.

Step 4: the termination URL is send

In case the request also contains the parameter `sap-sessioncmd` the termination URL is send. In this case the session mapping is removed from the HashMap and the internal session is invalidated, to free up memory.

Step 5: create the response and show the entries in the mapping table

The response just shows the mappings of the HashTable. This is done for monitoring reasons.

www.sdn.sap.com/irj/sdn/howtoguides